Improving pairwise coreference models through feature space hierarchy learning

Emmanuel Lassalle, Pascal Denis

Improving pairwise coreference models through feature space hierarchy learning

Emmanuel Lassalle
Alpage Project-team
INRIA & Univ. Paris Diderot
Sorbonne Paris Cité, F-75205 Paris
emmanuel.lassalle@ens-lyon.org

Pascal Denis
Magnet Project
INRIA Lille - Nord Europe
Avenue Heloïse, 59650 Villeneuve d'Ascq
pascal.denis@inria.fr

Abstract

This paper proposes a new method for significantly improving the performance of pairwise coreference models. Given a set of indicators, our method learns how to best separate types of mention pairs into equivalence classes for which we construct distinct classification models. In effect, our approach finds an optimal feature space (derived from a base feature set and indicator set) for discriminating coreferential mention pairs. Although our approach explores a very large space of possible feature spaces, it remains tractable by exploiting the structure of the hierarchies built from the indicators. Our experiments on the *CoNLL-2012 Shared Task* English datasets (gold mentions) indicate that our method is robust relative to different clustering strategies and evaluation metrics, showing large and consistent improvements over a single pairwise model using the same base features. Our best system obtains a competitive 67.2 of average F1 over MUC, B^3, and CEAF which, despite its simplicity, places it above the mean score of other systems on these datasets.

1 Introduction

Coreference resolution is the problem of partitioning a sequence of noun phrases (or *mentions*), as they occur in a natural language text, into a set of referential *entities*. A common approach to this problem is to separate it into two modules: on the one hand, one defines a model for evaluating coreference links, in general a discriminative classifier that detects coreferential mention pairs. On the other hand, one designs a method for grouping the detected links into a coherent global output (i.e. a partition over the set of entity mentions). This second step is typically achieved using greedy heuristics (McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002; Bengston and Roth, 2008), although more sophisticated clustering approaches have been used, too, such as cutting graph methods (Nicolae and Nicolae, 2006; Cai and Strube, 2010) and Integer Linear Programming (ILP) formulations (Klenner, 2007; Denis and Baldridge, 2009). Despite its simplicity, this two-step strategy remains competitive even when compared to more complex models utilizing a global loss (Bengston and Roth, 2008).

In this kind of architecture, the performance of the entire coreference system strongly depends on the quality of the local pairwise classifier.[1] Consequently, a lot of research effort on coreference resolution has focused on trying to boost the performance of the pairwise classifier. Numerous studies are concerned with feature extraction, typically trying to enrich the classifier with more linguistic knowledge and/or more world knowledge (Ng and Cardie, 2002; Kehler et al., 2004; Ponzetto and Strube, 2006; Bengston and Roth, 2008; Versley et al., 2008; Uryupina et al., 2011). A second line of work explores the use of distinct local models for different types of mentions, specifically for different types of *anaphoric* mentions based on their grammatical categories (such as pronouns, proper names, definite descriptions) (Morton, 2000; Ng, 2005; Denis and Baldridge, 2008).[2] An important justification for such spe-

[1] There are however no theoretical guarantees that improving pair classification will always result in overall improvements if the two modules are optimized independently.

[2] Sometimes, distinct sample selections are also adopted

cialized models is (psycho-)linguistic and comes from theoretical findings based on salience or accessibility (Ariel, 1988). It is worth noting that, from a machine learning point of view, this is related to feature extraction in that both approaches in effect recast the pairwise classification problem in higher dimensional feature spaces.

In this paper, we claim that mention pairs should not be processed by a single classifier, and instead should be handled through specific models. But we are furthermore interested in *learning* how to construct and select such differential models. Our argument is therefore based on statistical considerations, rather than on purely linguistic ones[3]. The main question we raise is, given a set of indicators (such as grammatical types, distance between two mentions, or named entity types), how to best partition the pool of mention pair examples in order to best discriminate coreferential pairs from non coreferential ones. In effect, we want to learn the "best" subspaces for our different models: that is, subspaces that are neither too coarse (i.e., unlikely to separate the data well) nor too specific (i.e., prone to data sparseness and noise). We will see that this is also equivalent to selecting a single large adequate feature space by using the data.

Our approach generalizes earlier approaches in important ways. For one thing, the definition of the different models is no longer restricted to grammatical typing (our model allows for various other types of indicators) or to the sole typing of the anaphoric mention (our models can also be specific to a particular type antecedent or to the two types of the mention pair). More importantly, we propose an original method for learning the best set of models that can be built from a given set of indicators and a training set. These models are organized in a hierarchy, wherein each leaf corresponds to a mutually disjoint subset of mention pair examples and the classifier that can be trained from it. Our models are trained using the *Online Passive-Aggressive* algorithm or *PA* (Crammer et al., 2006), a large margin version of the perceptron. Our method is exact in that it explores the full space of hierarchies (of size at least 2^{2^n}) definable on an indicator sequence, while remaining scalable by exploiting the particular structure of these hierarchies with dynamic programming. This approach also performs well, and it largely outperforms the single model. As will be shown based on a variety of experiments on the *CoNLL-2012 Shared Task* English datasets, these improvements are consistent across different evaluation metrics and for the most part independent of the clustering decoder that was used.

The rest of this paper is organized as follows. Section 2 discusses the underlying statistical hypotheses of the standard pairwise model and defines a simple alternative framework that uses a simple separation of mention pairs based on grammatical types. Next, in section 3, we generalize the method by introducing indicator hierarchies and explain how to learn the best models associated with them. Section 4 provides a brief system description and Section 5 evaluates the various models on CoNLL-2012 English datasets.

2 Modeling pairs

Pairwise models basically employ one local classifier to decide whether two mentions are coreferential or not. When using machine learning techniques, this involves certain assumptions about the statistical behavior of mention pairs.

2.1 Statistical assumptions

Let us adopt a probabilistic point of view to describe the prototype of pairwise models. Given a document, the number of mentions is fixed and each pair of mentions follows a certain distribution (that we partly observe in a feature space). The basic idea of pairwise models is to consider mention pairs independently from each other (that is why a decoder is necessary to enforce transitivity).

If we use a single classifier to process all pairs, then they are supposed to be identically distributed. We claim that pairs should not be processed by a single classifier because they are not identically distributed (or a least the distribution is too complex for the classifier); rather, we should separate different "types" on pairs and create a specific model for each of them.

Separating different kinds of pairs and handling them with different specific models can lead to more accurate global models. For instance, some coreference resolution systems process different kinds of anaphors separately, which suggests for example that pairs containing an anaphoric pronoun behave differently from pairs with non-

during the training of the distinct local models (Ng and Cardie, 2002; Uryupina, 2004).

[3] However it should be underlined that the statistical viewpoint is complementary to the linguistic work.

pronominal anaphors. One could rely on a rich set of features to capture complex distributions, but here we actually have a rather limited set of elementary features (see section 4) and, for instance, using products of features must be done carefully to avoid introducing noise in the model. Instead of imposing heuristic product of features, we will show that a clever separation of instances leads to significant improvements of the pairwise model.

2.2 Feature spaces

2.2.1 Definitions

We first introduce the problem more formally. Every pair of mentions m_i and m_j is modeled by a random variable:

$$\begin{array}{rcl} P_{ij} : \Omega & \rightarrow & \mathcal{X} \times \mathcal{Y} \\ \omega & \mapsto & (x_{ij}(\omega), y_{ij}(\omega)) \end{array}$$

where Ω classically represents randomness, $\mathcal{X}$ is the space of objects ("mention pairs") that is not directly observable and $y_{ij}(\omega) \in \mathcal{Y} = \{+1, -1\}$ are the labels indicating whether m_i and m_j are coreferential or not. To lighten the notations, we will not always write the index ij. Now we define a mapping:

$$\begin{array}{rcl} \phi_{\mathcal{F}} : \mathcal{X} & \rightarrow & \mathcal{F} \\ x & \mapsto & \mathbf{x} \end{array}$$

that casts pairs into a feature space $\mathcal{F}$ through which we observe them. For us, $\mathcal{F}$ is simply a vector space over $\mathbb{R}$ (in our case many features are Boolean; they are cast into $\mathbb{R}$ as 0 and 1).

For technical coherence, we assume that $\phi_{\mathcal{F}_1}(x(\omega))$ and $\phi_{\mathcal{F}_2}(x(\omega))$ have the same values when projected on the feature space $\mathcal{F}_1 \cap \mathcal{F}_2$: it means that common features from two feature spaces have the same values.

From this formal point of view, the task of coreference resolution consists in fixing $\phi_{\mathcal{F}}$, observing labeled samples $\{(\phi_{\mathcal{F}}(x), y)_t\}_{t \in TrainSet}$ and, given partially observed new variables $\{(\phi_{\mathcal{F}}(x))_t\}_{t \in TestSet}$, recovering the corresponding values of y.

2.2.2 Formalizing the statistical assumptions

We claimed before that all mention pairs seemed not to be identically distributed since, for example, pronouns do not behave like nominals. We can formulate this more rigorously: since the object space $\mathcal{X}$ is not directly observable, we do not know its complexity. In particular, when using a mapping to a too small feature space, the classifier cannot capture the distribution very well: the data is too noisy.

Now if we say that pronominal anaphora do not behave like other anaphora, we distinguish two kinds of pair i.e. we state that the distribution of pairs in $\mathcal{X}$ is a mixture of two distributions, and we deterministically separate pairs to their specific distribution part. In this way, we may separate positive and negative pairs more easily if we cast each kind of pair into a specific feature space. Let us call these feature spaces $\mathcal{F}_1$ and $\mathcal{F}_2$. We can either create two independent classifiers on $\mathcal{F}_1$ and $\mathcal{F}_2$ to process each kind of pair or define a single model on a larger feature space $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2$. If the model is linear (which is our case), these approaches happen to be equivalent.

So we can actually assume that the random variables P_{ij} are identically distributed, but drawn from a complex mixture. A new issue arises: we need to find a mapping $\phi_{\mathcal{F}}$ that renders the best view on the distribution of the data.

From a theoretical viewpoint, the higher the dimension of the feature space (imagine taking the direct sum of all feature spaces), the more we get details on the distribution of mention pairs and the more we can expect to separate positives and negatives accurately. In practice, we have to cope with data sparsity: there will not be enough data to properly train a linear model on such a space. Finally, we seek a feature space situated between the two extremes of a space that is too big (sparseness) or too small (noisy data). The core of this work is to define a general method for choosing the most adequate space $\mathcal{F}$ among a huge number of possibilities when we do not know *a priori* which is the best.

2.2.3 Linear models

In this work, we try to linearly separate positive and negative instances in the large space $\mathcal{F}$ with the *Online Passive-Aggressive* (PA) algorithm (Crammer et al., 2006): the model learns a parameter vector $\mathbf{w}$ that defines a hyperplane that cuts the space into two parts. The predicted class of a pair x with feature vector $\phi_{\mathcal{F}}(x)$ is given by:

$$C_{\mathcal{F}}(x) := sign(\mathbf{w}^T \cdot \phi_{\mathcal{F}}(x))$$

Linearity implies an equivalence between: (i) separating instances of two types, t_1 and t_2, in two

independent models with respective feature spaces $\mathcal{F}_1$ and $\mathcal{F}_2$ and parameters $\mathbf{w}^1$ and $\mathbf{w}^2$, and (ii) a single model on $\mathcal{F}_1 \oplus \mathcal{F}_2$. To see why, let us define the map:

$$\phi_{\mathcal{F}_1 \oplus \mathcal{F}_2}(x) := \begin{cases} \begin{pmatrix} \phi_{\mathcal{F}_1}(x)^T & 0 \end{pmatrix}^T & \text{if } x \text{ typed } t_1 \\ \begin{pmatrix} 0 & \phi_{\mathcal{F}_2}(x)^T \end{pmatrix}^T & \text{if } x \text{ typed } t_2 \end{cases}$$

and the parameter vector $\mathbf{w} = \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \end{pmatrix} \in \mathcal{F}_1 \oplus \mathcal{F}_2$. Then we have:

$$C_{\mathcal{F}_1 \oplus \mathcal{F}_2}(x) = \begin{cases} C_{\mathcal{F}_1}(x) & \text{if } x \text{ typed } t_1 \\ C_{\mathcal{F}_2}(x) & \text{if } x \text{ typed } t_2 \end{cases}$$

Now we check that the same property applies when the PA fits its parameter $\mathbf{w}$. For each new instance of the training set, the weight is updated according to the following rule[4]:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathcal{F}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 \ s.t. \ l(\mathbf{w}; (x_t, y_t)) = 0$$

where $l(\mathbf{w}; (x_t, y_t)) = \min(0, 1 - y_t(\mathbf{w} \cdot \phi_{\mathcal{F}}(x_t)))$, so that when $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2$, the minimum if x is typed t_1 is $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_{t+1} \\ \mathbf{w}^2_t \end{pmatrix}$ and if x is typed t_2 is $\mathbf{w}_{t+1} = \begin{pmatrix} \mathbf{w}^1_t \\ \mathbf{w}^2_{t+1} \end{pmatrix}$ where the $\mathbf{w}^i_{t+1}$ correspond to the updates in space $\mathcal{F}_i$ independently from the rest. This result can be extended easily to the case of n feature spaces. Thus, with a deterministic separation of the data, a large model can be learned using smaller independent models.

2.3 An example: separation by *gramtype*

To motivate our approach, we first introduce a simple separation of mention pairs which creates 9 models obtained by considering all possible pairs of grammatical types {*nominal, name, pronoun*} for *both* mentions in the pair (a similar fine-grained separation can be found in (Chen et al., 2011)). This is equivalent to using 9 different feature spaces $\mathcal{F}_1, \ldots, \mathcal{F}_9$ to capture the global distribution of pairs. With the PA, this is also a single model with feature space $\mathcal{F} = \mathcal{F}_1 \oplus \cdots \oplus \mathcal{F}_9$. We will call it the GRAMTYPE model.

As we will see in Section 5, these separated models significantly outperform a single model that uses the same base feature set. But we would like to define a method that adapts a feature space to the data by choosing the most adequate separation of pairs.

3 Hierarchizing feature spaces

In this section, we have to keep in mind that separating the pairs in different models is the same as building a large feature space in which the parameter $\mathbf{w}$ can be learned by parts in independent subspaces.

3.1 Indicators on pairs

For establishing a structure on feature spaces, we use *indicators* which are deterministic functions on mention pairs with a small number of outputs. Indicators classify pairs in predefined categories in one-to-one correspondence with independent feature spaces. We can reuse some features of the system as indicators, e.g. the grammatical or named entity types. We can also employ functions that are not used as features, e.g. the approximate position of one of the mentions in the text.

The small number of outputs of an indicator is required for practical reasons: if a category of pairs is too refined, the associated feature space will suffer from data sparsity. Accordingly, distance-based indicators must be approximated by coarse histograms. In our experiments the outputs never exceeded a dozen values. One way to reduce the output span of an indicator is to binarize it like binarizing a tree (many possible binarizations). This operation produces a hierarchy of indicators which is exactly the structure we exploit in what follows.

3.2 Hierarchies for separating pairs

We define hierarchies as combinations of indicators creating finer categories of mention pairs: given a finite sequence of indicators, a mention pair is classified by applying the indicators successively, each time refining a category into subcategories, just like in a decision tree (each node having the same number of children as the number of outputs of its indicator). We allow the classification to stop before applying the last indicator, but the behavior must be the same for all the instances. So a hierarchy is basically a sub-tree of the complete decision tree that contains copies of the same indicator at each level.

If all the leaves of the decision tree have the

[4]The parameter is updated to obtain a margin of a least 1. It does not change if the instance is already correctly classified with such margin.

same depth, this corresponds to taking the Cartesian product of outputs of all indicators for indexing the categories. In that case, we refer to *product-hierarchies*. The GRAMTYPE model can be seen as a two level product-hierarchy (figure 1).

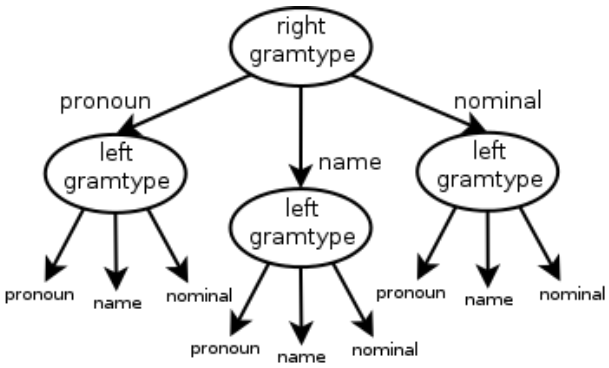

Figure 1: GRAMTYPE seen as a product-hierarchy

Product-hierarchies will be the starting point of our method to find a feature space that fits the data.

Now choosing a relevant sequence of indicators should be achieved through linguistic intuitions and theoretical work (*gramtype* separation is one of them). The system will find by itself the best usage of the indicators when optimizing the hierarchy. The sequence is a parameter of the model.

3.3 Relation with feature spaces

Like we did for the GRAMTYPE model, we associate a feature space $\mathcal{F}_i$ to each leaf of a hierarchy. Likewise, the sum $\mathcal{F} = \bigoplus_i \mathcal{F}_i$ defines a large feature space. The corresponding parameter $\mathbf{w}$ of the model can be obtained by learning the $\mathbf{w}_i$ in $\mathcal{F}_i$.

Given a sequence of indicators, the number of different hierarchies we can define is equal to the number of sub-trees of the complete decision tree (each non-leaf node having all its children). The minimal case is when all indicators are Boolean. The number of full binary trees of height at most n can be computed by the following recursion: $T(1) = 1$ and $T(n+1) = 1 + T(n)^2$. So $T(n) \geq 2^{2^n}$: even with small values of n, the number of different hierarchies (or large feature spaces) definable with a sequence of indicators is gigantic (e.g. $T(10) \approx 3.8.10^{90}$).

Among all the possibilities for a large feature space, many are irrelevant because for them the data is too sparse or too noisy in some subspaces. We need a general method for finding an adequate space without enumerating and testing each of them.

3.4 Optimizing hierarchies

Let us assume now that the sequence of indicators is fixed, and let n be its length. To find the best feature space among a very high number of possibilities, we need a criterion we can apply without too much additional computation. For that we only evaluate the feature space locally on pairs, i.e. without applying a decoder on the output. We employ 3 measures on pairwise classification results: precision, recall and F1-score. Now selecting the best space for one of these measures can be achieved by using dynamic programming techniques. In the rest of the paper, we will optimize the F1-score.

Training the hierarchy Starting from the product-hierarchy, we associate a classifier and its proper feature space to each node of the tree[5]. The classifiers are then trained as follows: for each instance there is a unique path from the root to a leaf of the complete tree. Each classifier situated on the path is updated with this instance. The number of iterations of the Passive-Aggressive is fixed.

Computing scores After training, we test all the classifiers on another set of pairs[6]. Again, a classifier is tested on an instance only if it is situated on the path from the root to the leaf associated with the instance. We obtain TP/FP/FN numbers[7] on pair classifications that are sufficient to compute the F1-score. As for training, the data on which a classifier at a given node is evaluated is the same as the union of all data used to evaluate the classifiers corresponding to the children of this node. Thus we are able to compare the scores obtained at a node to the "union of the scores" obtained at its children.

Cutting down the hierarchy For the moment we have a complete tree with a classifier at each node. We use a dynamic programming technique to compute the best hierarchy by cutting this tree and only keeping classifiers situated at the leaf. The algorithm assembles the best local models (or feature spaces) together to create larger models. It goes from the leaves to the root and cuts the subtree starting at a node whenever it does not pro-

[5] In the experiments, the classifiers use a copy of a same feature space, but not the same data, which corresponds to crossing the features with the categories of the decision tree.

[6] The training set is cut into two parts, for training and testing the hierarchy. We used 10-fold cross-validation in our experiments.

[7] True positives, false positives and false negatives.

vide a better score than the node itself, or on the contrary propagates the score of the sub-tree when there is an improvement. The details are given in algorithm 1.

```
1 list ← list of nodes given by a breadth-first
  search for node in reversed list do
2     if node.children ≠ ∅ then
3         if sum-score(node.children) >
          node.score then
4             node.TP/FP/FN ←
              sum-num(node.children)
5         else
6             node.children ← ∅
7         end
8     end
9 end
```

Algorithm 1: Cutting down a hierarchy

Let us briefly discuss the correctness and complexity of the algorithm. Each node is seen two times so the time complexity is linear in the number of nodes which is at least $\mathcal{O}(2^n)$. However, only nodes that have encountered at least one training instance are useful and there are $\mathcal{O}(n \times k)$ such nodes (where k the size of the training set). So we can optimize the algorithm to run in time $\mathcal{O}(n \times k)$[8]. If we scan the list obtained by breadth-first search backwards, we are ensured that every node will be processed after its children. (*node.children*) is the set of children of *node*, and (*node.score*) its score. *sum-num* provides TP/FP/FN by simply adding those of the children and *sum-score* computes the score based on these new TP/FP/FN numbers. (line 6) cuts the children of a node when they are not used in the best score. The algorithm thus propagates the best scores from the leaves to the root which finally gives a single score corresponding to the best hierarchy. Only the leaves used to compute the best score are kept, and they define the best hierarchy.

Relation between cutting and the global feature space We can see the operation of cutting as replacing a group of subspaces by a single subspace in the sum (see figure 2). So cutting down the product-hierarchy amounts to reducing the global initial feature space in an optimal way.

[8]In our experiments, cutting down the hierarchy was achieved very quickly, and the total training time was about five times longer than with a single model.

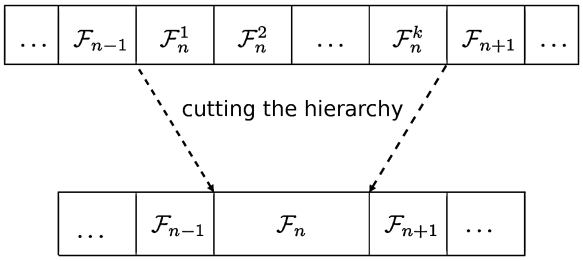

Figure 2: Cutting down the hierarchy reduces the feature space

To sum up, the whole procedure is equivalent to training more than $\mathcal{O}(2^n)$ perceptrons simultaneously and selecting the best performing.

4 System description

Our system consists in the pairwise model obtained by cutting a hierarchy (the PA with selected feature space) and using a greedy decoder to create clusters from the output. It is parametrized by the choice of the initial sequence of indicators.

4.1 The base features

We used classical features that can be found in details in (Bengston and Roth, 2008) and (Rahman and Ng, 2011): grammatical type and subtype of mentions, string match and substring, apposition and copula, distance (number of separating mentions/sentences/words), gender/number match, synonymy/hypernym and animacy (using WordNet), family name (based on lists), named entity types, syntactic features (gold parse) and anaphoricity detection.

4.2 Indicators

As indicators we used: left and right grammatical types and subtypes, entity types, a boolean indicating if the mentions are in the same sentence, and a very coarse histogram of distance in terms of sentences. We systematically included right gramtype and left gramtype in the sequences and added other indicators, producing sequences of different lengths. The parameter was optimized by document categories using a development set *after* decoding the output of the pairwise model.

4.3 Decoders

We tested 3 classical greedy link selection strategies that form clusters from the classifier decision: Closest-First (merge mentions with their closest coreferent mention on the left) (Soon et al., 2001),

Best-first (merge mentions with the mention on the left having the highest positive score) (Ng and Cardie, 2002; Bengston and Roth, 2008), and Aggressive-Merge (transitive closure on positive pairs) (McCarthy and Lehnert, 1995). Each of these decoders is typically (although not always) used in tandem with a specific sampling selection at training. Thus, Closest-First for instance is used in combination with a sample selection that generates training instances only for the mentions that occur between the closest antecedent and the anaphor (Soon et al., 2001).

	P	R	F1
SINGLE MODEL	22.28	**63.50**	32.99
RIGHT-TYPE	29.31	45.23	35.58
GRAMTYPE	39.12	45.83	42.21
BEST HIERARCHY	**45.27**	51.98	**48.40**

Table 1: Pairwise scores on CoNLL-2012 test.

5 Experiments

5.1 Data

We evaluated the system on the English part of the corpus provided in the *CoNLL-2012 Shared Task* (Pradhan et al., 2012), referred to as CoNLL-2012 here. The corpus contains 7 categories of documents (over 2K documents, 1.3M words). We used the official train/dev/test data sets. We evaluated our system in the *closed mode* which requires that only provided data is used.

5.2 Settings

Our baselines are a SINGLE MODEL, the GRAMTYPE model (section 2) and a RIGHT-TYPE model, defined as the first level of the *gramtype* product hierarchy (i.e. grammatical type of the anaphora (Morton, 2000)), with each greedy decoder and also the original sampling with a single model associated with those decoders.

The hierarchies were trained with 10-fold cross-validation on the training set (the hierarchies are cut after cumulating the scores obtained by cross-validation) and their parameters are optimized *by document category* on the development set: the sequence of indicators obtaining the best average score *after* decoding was selected as parameter for the category. The obtained hierarchy is referred to as the BEST HIERARCHY in the results. We fixed the number of iterations for the PA for all models.

In our experiments, we consider only the *gold mentions*. This is a rather idealized setting but our focus is on comparing various pairwise local models rather than on building a full coreference resolution system. Also, we wanted to avoid having to consider too many parameters in our experiments.

5.3 Evaluation metrics

We use the three metrics that are most commonly used[9], namely:

MUC (Vilain et al., 1995) computes for each true entity cluster the number of system clusters that are needed to cover it. Precision is this quantity divided by the true cluster size minus one. Recall is obtained by reversing true and predicated clusters. F1 is the harmonic mean.

B^3 (Bagga and Baldwin, 1998) computes recall and precision scores for each mention, based on the intersection between the system/true clusters for that mention. Precision is the ratio of the intersection and the true cluster sizes, while recall is the ratio of the intersection to the system cluster sizes. Global recall, precision, and F1 scores are obtained by averaging over the mention scores.

CEAF (Luo, 2005) scores are obtained by computing the best one-to-one mapping between the system/true partitions, which is equivalent to finding the best optimal alignment in the bipartite graph formed out of these partitions. We use the ϕ_4 similarity function from (Luo, 2005).

These metrics were recently used in the *CoNLL-2011* and *-2012 Shared Tasks*. In addition, these campaigns use an unweighted average over the F1 scores given by the three metrics. Following common practice, we use micro-averaging when reporting our scores for entire datasets.

5.4 Results

The results obtained by the system are reported in table 2. The original sampling for the single model associated to Closest-First and Best-First decoder are referred to as SOON and NGCARDIE.

The P/R/F1 pairwise scores before decoding are given in table 1. BEST HIERARCHY obtains a strong improvement in F1 (+15), a better precision and a less significant diminution of recall compared to GRAMTYPE and RIGHT-TYPE.

[9]BLANC metric (Recasens and Hovy, 2011) results are not reported since they are not used to compute the CoNLL-2012 global score. However we can mention that in our experiments, using hierarchies had a positive effect similar to what was observed on B^3 and CEAF.

	MUC			B^3			CEAF			
Closest-First	P	R	F1	P	R	F1	P	R	F1	Mean
SOON	79.49	**93.72**	**86.02**	26.23	**89.43**	40.56	**49.74**	19.92	28.44	51.67
SINGLE MODEL	78.95	75.15	77.0	51.88	68.42	59.01	37.79	43.89	40.61	58.87
RIGHT-TYPE	79.36	67.57	72.99	69.43	56.78	62.47	41.17	**61.66**	49.37	61.61
GRAMTYPE	80.5	71.12	75.52	66.39	61.04	63.6	43.11	59.93	50.15	63.09
BEST HIERARCHY	**83.23**	73.72	78.19	**73.5**	67.09	**70.15**	47.3	60.89	**53.24**	**67.19**

	MUC			B^3			CEAF			
Best-First	P	R	F1	P	R	F1	P	R	F1	Mean
NGCARDIE	**81.02**	**93.82**	**86.95**	23.33	**93.92**	37.37	**40.31**	18.97	25.8	50.04
SINGLE MODEL	79.22	73.75	76.39	40.93	75.48	53.08	30.52	37.59	33.69	54.39
RIGHT-TYPE	77.13	65.09	70.60	48.11	66.21	55.73	31.07	47.30	37.50	54.61
GRAMTYPE	77.21	65.89	71.1	49.77	67.19	57.18	32.08	**47.83**	38.41	55.56
BEST HIERARCHY	78.11	69.82	73.73	**53.62**	70.86	**61.05**	35.04	46.67	**40.03**	**58.27**

	MUC			B^3			CEAF			
Aggressive-Merge	P	R	F1	P	R	F1	P	R	F1	Mean
SINGLE MODEL	83.15	**88.65**	**85.81**	35.67	**88.18**	50.79	36.3	28.27	31.78	56.13
RIGHT-TYPE	83.48	89.79	86.52	36.82	88.08	51.93	45.30	33.84	38.74	59.07
GRAMTYPE	83.12	84.27	83.69	44.73	81.58	57.78	45.02	42.94	43.95	61.81
BEST HIERARCHY	**83.26**	85.2	84.22	**45.65**	82.48	**58.77**	**46.28**	**43.13**	**44.65**	**62.55**

Table 2: CoNLL-2012 test (gold mentions): Closest-First, Best-First and Aggressive-Merge decoders.

Despite the use of greedy decoders, we observe a large positive effect of pair separation in the pairwise models on the outputs. On the mean score, the use of distinct models versus a single model yields F1 increases from 6.4 up to 8.3 depending on the decoder. Irrespective of the decoder being used, GRAMTYPE always outperforms RIGHT-TYPE and single model and is always outperformed by BEST HIERARCHY model.

Interestingly, we see that the increment in pairwise and global score are not proportional: for instance, the strong improvement of F1 between RIGHT-TYPE and GRAMTYPE results in a small amelioration of the global score.

Depending on the document category, we found some variations as to which hierarchy was learned in each setting, but we noticed that parameters starting with right and left gramtypes often produced quite good hierarchies: for instance right gramtype $\rightarrow$ left gramtype $\rightarrow$ same sentence $\rightarrow$ right named entity type.

We observed that product-hierarchies did not performed well without cutting (especially when using longer sequences of indicators, because of data sparsity) and could obtain scores lower than the single model. Hopefully, after cutting them the results always became better as the resulting hierarchy was more balanced.

Looking at the different metrics, we notice that overall, pair separation improves B^3 and CEAF (but not always MUC) after decoding the output: GRAMTYPE provides a better mean score than the single model, and BEST HIERARCHY gives the highest B^3, CEAF and mean score.

The best classifier-decoder combination reaches a score of 67.19, which would place it above the mean score (66.41) of the systems that took part in the *CoNLL-2012 Shared Task* (gold mentions track). Except for the first at 77.22, the best performing systems have a score around 68-69. Considering the simple decoding strategy we employed, our current system sets up a strong baseline.

6 Conclusion and perspectives

In this paper, we described a method for selecting a feature space among a very large number of choices by using linearity and by combining indicators to separate the instances. We employed dynamic programming on hierarchies of indicators to compute the feature space providing the best pairwise classifications efficiently. We applied this

method to optimize the pairwise model of a coreference resolution system. Using different kinds of greedy decoders, we showed a significant improvement of the system.

Our approach is flexible in that we can use a variety of indicators. In the future we will apply the hierarchies on finer feature spaces to make more accurate optimizations. Observing that the general method of cutting down hierarchies is not restricted to modeling mention pairs, but can be applied to problems having Boolean aspects, we aim at employing hierarchies to address other tasks in computational linguistics (e.g. anaphoricity detection or discourse and temporal relation classification wherein position information may help separating the data).

In this work, we have only considered standard, heuristic linking strategies like Closest-First. So, a natural extension of this work is to combine our method for learning pairwise models with more sophisticated decoding strategies (like *Bestcut* or using ILP). Then we can test the impact of hierarchies with more realistic settings.

Finally, the method for cutting hierarchies should be compared to more general but similar methods, for instance polynomial kernels for SVM and tree-based methods (Hastie et al., 2001). We also plan to extend our method by breaking the symmetry of our hierarchies. Instead of cutting product-hierarchies, we will employ usual techniques to build decision trees[10] and apply our cutting method on their structure. The objective is twofold: first, we will get rid of the sequence of indicators as parameter. Second, we will avoid fragmentation or overfitting (which can arise with classification trees) by deriving an optimal large margin linear model from the tree structure.

Acknowledgments

We thank the *ACL 2013* anonymous reviewers for their valuable comments.

References

M. Ariel. 1988. Referring and accessibility. *Journal of Linguistics*, pages 65–87.

A. Bagga and B. Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of LREC 1998*, pages 563–566.

Mohit Bansal and Dan Klein. 2012. Coreference semantics from web features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 389–398. Association for Computational Linguistics.

Eric Bengston and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of EMNLP 2008*, pages 294–303, Honolulu, Hawaii.

Jie Cai and Michael Strube. 2010. End-to-end coreference resolution via hypergraph partitioning. In *COLING*, pages 143–151.

Bin Chen, Jian Su, Sinno Jialin Pan, and Chew Lim Tan. 2011. A unified event coreference resolution by integrating multiple resolvers. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 102–110, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of EMNLP 2008*, pages 660–669, Honolulu, Hawaii.

Pascal Denis and Jason Baldridge. 2009. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 43.

Trevor Hastie, Robert Tibshirani, and J. H. Friedman. 2001. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag.

A. Kehler, D. Appelt, L. Taylor, and A. Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of HLT-NAACL 2004*.

M. Klenner. 2007. Enforcing coherence on coreference sets. In *Proceedings of RANLP 2007*.

X. Luo. 2005. On coreference resolution performance metrics. In *Proceedings of HLT-NAACL 2005*, pages 25–32.

J. F. McCarthy and W. G. Lehnert. 1995. Using decision trees for coreference resolution. In *IJCAI*, pages 1050–1055.

T. Morton. 2000. Coreference for NLP applications. In *Proceedings of ACL 2000*, Hong Kong.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of ACL 2002*, pages 104–111.

[10] (Bansal and Klein, 2012) show good performances of decision trees on coreference resolution.

V. Ng. 2005. Supervised ranking for pronoun resolution: Some recent improvements. In *Proceedings of AAAI 2005*.

Cristina Nicolae and Gabriel Nicolae. 2006. Bestcut: A graph algorithm for coreference resolution. In *EMNLP*, pages 275–283.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the HLT 2006*, pages 192–199, New York City, N.Y.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2011. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *J. Artif. Int. Res.*, 40(1):469–521.

Recasens and Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17:485–510, 9.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Olga Uryupina, Massimo Poesio, Claudio Giuliano, and Kateryna Tymoshenko. 2011. Disambiguation and filtering methods in using web knowledge for coreference resolution. In *FLAIRS Conference*.

O. Uryupina. 2004. Linguistically motivated sample selection for coreference resolution. In *Proceedings of DAARC 2004*, Furnas.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *COLING*, pages 961–968.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings fo the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, CA. Morgan Kaufmann.